Gnu Privacy Guard (GnuPG) mini HOWTO

J.S.A.A.F. de Winter (English)Brenno

brenno (at) dewinter.com

Fischer v. Mollard (German)Michael

fischer (at) math.uni-goettingen.de

BaartArjen

arjen (at) andromeda.nl

도운이

조성표

koseph (at) netian.com

이 문서는 GNU Privacy Guard(GnuPG, GNU 사생활 경비원)의 사용법을 설명하고 있다. GnuPG는 오픈 소스로 진행하고 있는 OpenPGP와 호환성 있는 암호화 시스템이다. 이 프로그램을 완전히 자유롭게 하기 위하여 RSA 알고리즘과 다른 특허등록된 알고리즘은 피했다. 원래 문서는 독일의 마이클 피셔 v. 몰라드씨가 작성하였다. 그 분이 적은 글을 번역했고 몇 가지를 조정하였으므로 완전히 동일한 사본으로 간주할 수는 없다.

고친 과정

고침 0.1.42003-03-06고친이 koseph
오리지널 0.1.3 영문 SGML문서를 KLDP 프로젝트에 맞도록 수정(한국어판)
고침 0.1.32002-05-17고친이
몇 가지 수정(독일어판)

고침 0.1.22002-04-03고친이
몇 가지 수정, 프런트 엔트 부분 추가(영문판)
고침 0.1.1알 수 없음 고친이
몇 가지 수정, 보안의 경계선 부분 추가(독일어판)
고침 0.1 1999-04-30고친이
독일어 원본을 영문으로 최초 작성

1. 소개의 글

본 문서는 브레노 J.S.A.A.F. 드 윈터씨가 작성한 영문 문서를 번역한 것으로, KLDP 프로젝트 (http://kldp.org/)에서 요구하는 SGML 형식으로 변환한 것입니다. 작업 초기 SGML로 작업해야 한다는 걸 모르고 HTML로 작업했다가 다시 SGML로 작업해서 시간이 곱절 걸린 번역본입니다. 아무쪼록 도움이 되시길.....

1.1. 저작권 정보

Copyright (C) 1999 Michael Fischer v. Mollard (original German version)

Copyright (C) 1999 Brenno J.S.A.A.F. de Winter (English)

Copyright (C) 2002 Arjen Baart (Dutch version)

Copyright (C) 2003 조성표 (한국어판)

본 문서는 여러분이 자유 소프트웨어 재단(Free Software Foundation)에서 발행한 GNU 라이브러리 일반 공개 사용권(GNU Library General Public License) (http://www.gnu.org/copyleft/fdl.html)의 규정(즉 사용권 2판이나 혹은 여러분의 선택에 따라 더 이후의 사용권)을 준수하는 한 재유포하거나 수정할 수 자유 문서입니다. 단, 저작권에 대한 본 사항을 꼭 명시하여야 합니다.

여러분은 이 라이브러리와 함께 GNU 라이브러리 일반 공개 사용권의 사본을 꼭 받아 보아야만 합니다. 만일 그렇지 않다면 다음의 주소로 연락하기 바랍니다: Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

1.2. 책임의 한계

본 라이브러리는 이것이 유용하길 바라는 마음에서 배포합니다만, 어떠한 보증도 없습니다. 상업적인 용도나 특정한 목적에 맞게 사용 했을때 의무적으로 따라 붙는 보증도 없습니다. 더 자세한 내용은 GNU 라이브러리 일반 대중 사용권에 대해 보기 바랍니다.

본 문서의 집필자 및 역자는 문서의 내용이 야기할 수 있는 어떠한 결과에 대해서도 책임을 지지 않습니다. 본 문서에서 내포하고 있는 정보 및 예제는 여러분이 알아서 활용하십시오. 비록 최선을 다했으나 번역이 갖는 고유한 특성과 역자의 실력부족으로 인하여 이 문서는 틀린 점이나 오류가 있을 수도 있습니다.

1.3. 감사의 글

가장 먼저 늘 자신을 아끼지 않는, 나를 아끼는 사람에게 감사 드립니다.

그리고, 앞으로 리눅스를 좋아 하시는 분, 좋아 해보려고 하는 분, 이런 번역 작업을 해 보시려고 하는 분들이 늘어났으면 하고 바랍니다. :)

1.4. 피드백

이 문서를 읽을 때 오역이나 오탈자가 있다면 너그러이 이해하고 수정사항이 있다면 <koseph (at) netian.com>으로 수정사항을 보내거나, 스스로 수정하여 가용한 사이트에 제출하거나, 둘다 하거나 알아서 하기 바랍니다.

2. 개념

2.1. 공개키 암호화

암호화 하는데 있어 고전적인 방법은 단 하나의 키만을 사용한다. 보내는 사람(송신자)은 이 키를 이용하여 메시지를 암호화한다. 암호화한 메시지를 풀기(복호화 혹은 평문화) 위해서 받는 사람(수신자)은 보낸 사람이 사용했던 것과 똑같은 키를 가지고 있을 필요가 있다. 이 키를 받는 사람에게 어떤 방법을 이용하여 전달하여야 하는데, 그것은 다른 사람들에게 이 키를 얻을 기회를 주지 않는 것이다. 만약 어떤 다른 사람이 이 키를 갖게 된다면, 이 암호화 방법은 쓸모없는 것이다.

소위 공개키(public key)라는 것을 이용하면 이러한 문제를 해결할 수 있다. 공개키는 두 개의 키가 관여하는 개념이다. 키 한개는 공개키로 모든 부류의 매체를 통해 유포할 수 있고 어느 누구든 가질 수 있다. 나머지 키 한개는 비밀키(private key)이다. 이 키는 비밀스러운 것으로 유포해서는 안된다. 이 비밀키는 오로지 키의 주인(소유자)만이 사용할 수 있다. 구현이 잘된 시스템인 경우 비밀키는 공개키를 이용해 추측할 수 없다. 이제 보내는 사람이 받는 사람의 공개키를 이용해 메시지를 암호화하여 보낼 것이다. 그런 후에 받는 사람은 자신의 비밀키를 이용하여 암호화된 메시지를 풀어보게 되는 것이다.

이러한 개념에서 중요한 것은 비밀키는 철저히 비밀을 지켜야하고(보안기밀을 유지해야 하고) 비밀키의 주인을 제외하곤 타인에게 주거나 남들이 입수하도록 하면 안된다는 것이다. 여러분은 여러분의 비밀키를 인터넷을 통해 전송해서는 안된다. 또한 텔넷(**telnet**)을 통해 GnuPG를 사용하는 것은 대단히 현명하지 않은 행동이다. (여러분은 보안상 위험성이 높은 텔넷을 이용하는 것을 절대 쓰지 않는 것을 고려해 봄직하다)

2.2. 디지틀 서명

메시지가 정말로 틀림없이 보낸 그 사람이 보낸 것인가를 보여주기 위해서 디지틀 서명이라는 개념이 나왔다. 말 그대로 보내는 사람이 메시지를 디지틀을 이용하여 서명하는 것이다. 이 서명을 이용하므로서 여러분은 메시지의 신빙성(authenticity)를 점검할 수 있다. 이를 이용하므써 트로이 목마(어떤 문제를 고칠 수 있다고 하지만 실제로는 바이러스나 여러분의 컴퓨터에 있는 데이터에 나쁜 짓거리를 하는 메시지)에 대한 위험을 줄일 수 있다. 또한 정보 혹은 데이터를 보낸 곳이 진짜 출처인지를 확인할 수 있어서 그 진위를 구별할 수 있다.

디지틀 서명은 특정인의 비밀키와 사용한 텍스트를 조합하여 만든다. 보낸 사람의 공개키를 이용하면 메시지를 확인할 수 있다. 보낸 사람이 맞는지 만을 점검하는 것이 아니고 보낸 내용까지도 점검한다. 그러므로 여러분은 보낸 사람으로부터 그 메시지가 왔고 전송 도중에 메시지의 내용이 변하지 않았음을 아는 것이다.

2.3. 신용 웹

공개키 알고리즘의 취약점은 공개키를 유포하는데 있다. 어떤 사용자가 널리 알려진 사용자 신분을 위장하여 공개키를 가져갈 수 있다. 만약 이 특정한 키를 이용하여 메시지를 암호화하는 경우, 침입자는 메시지를 풀어서 읽을 수 있다. 만약 침입자가 이를 손에 넣게 되면 여전히 실제 받는 사람에게 진짜 공개키로 암호화하므로, 이러한 공격에 대해선 알아차릴 수 없다.

PGP 솔루션(그리고 자동적으로 GnuPG 솔루션이기 때문에)은 서명하는 코드에 존재한다. 공개키는 다른 사람이 서명할 수 있다. 이 서명은 UID(User IDentification의 약자, 사용자 신분)를 이용한 키가 실제로 그 사람이라고 주장하는 사람의 것인지를 알려준다. 그런 다음에 나머지는 GnuPG를 사용자가 그 서명을 얼마나 믿는가에 달려 있다. 여러분은 여러분이 키를 보낸 사람을 믿을때 키가 신뢰하고 그 키가 실제로 그 사람 것이라는 것도 확실히 안다. 오로지 여러분이 서명한 사람의 키를 신뢰할 때만, 여러분은 그 서명을 신뢰할 수 있다. 확실하게 그 키가 맞는 것임을 확신하려면 여러분은 절대적인 신뢰를 주기 전에 믿을 수 있는 채널을 통해 지문(finger print)이 같은지 비교해 봐야만 한다.

2.4. 보안에 대한 경계선

만약 여러분이 비밀로 하고픈 데이터를 가지고 있다면, 단지 어떤 인코딩 알고리즘을 쓸 것인지를 결정하는 것만이 전부가 아니다. 여러분의 시스템 보안에 대해 일반적으로 생각해야만 한다. 기본적으로 우리는 PGP는 안전하다고 생각하고 이 문서를 작성하고 있는 지금 필자가 아는 PGP 크랙 사고는 없다. 하지만 이러한 사실이 모든 인코딩 방법이 안전한 것이라는 것을 의미하진 않는다. (예를 들어 안기부가 어찌어찌 해서 PGP를 크랙한 경우 필자에게 이를 알리지 않을 것이고, 정말 악의적인 이유로 크랙을 한 사람 역시 그럴것이다). 하지만, 설사 PGP가 완벽하게 '해킹할 수 없는' 것이라 할지라도 안전을 위협하는 다른 방법을 쓸 수도 있다. 1999년 2월초 하드디스크 상에서 비밀 PGP키를 찾아서 찾은 키를 FTP로 전송하는 트로이 목마가 발견됐다. 만약 좋지 않은 암호를 선택한 경우라면 비밀키는 쉽게 크랙당할 수 있다.

또 다른 기술적인 가능성(비록 더 어렵긴 하지만)은 키보드 입력내용을 전파하는 트로이 목마다. 또한 화면의 내용까지 보낼 가능성(하지만 매우 어려운 일이다)도 있다. 이런 방법을 이용하면 스크램블된 메시지를 크래킹할 필요가 없다. 이러한 모든 위험에 대비하여 실제로 운영할 훌륭하고 깊이 생각한 보안 계획이 필요한 것이다.

사람들 사이에 편집증을 만들기 보다는, 보다 안전하기 위해선 필요한 것이 더 있다는 것을 지적하기 위함이다. 가장 중요한 것은 암호화는 보안에 있어 첫발이라는 것과 모든 것을 해결해 주는 것이 아니라는 것을 깨닫는 것이다. 1999년 3월에 멜리사 바이러스에 출현한 트로이 목마는 많은 기업이 이러한 것에 대비하지 않았음을 증명했다.

3. 설치

3.1. GnuPG 소스

공식적인 다운로드 사이트는 GnuPG 홈페이지 (http://www.gnupg.org/download.html)이다. 홈페이지에 가보면 미러 사이트에 대한 링크를 찾을 수 있다. 법적인 제약으로 인하여 미합중국에 기반을 두고 있는 서버에서 GnuPG를 다운로드 하는 것을 허가하지 않고 있다. 미합중국은 암호화 소프트웨어의 수출제한을 강요하고 있다. 이것이 PGP가 항상 국제판과 국내(미합중국용)으로 나와 있는 이유이다. 국제판의 경우에 소스 코드는 책과 같은 인쇄물로 전달해왔다. 유럽(오슬로)에서 스캔해 왔다. 국제 PGP 홈페이지 (http://www.pgpi.com)에서 더 많은 정보를 찾을 수 있다. PGP 국제판은 다시 재수출 되지 않는 한 미합중국내로 들여오는 것은 자유다.

만약 여러분이 설치된 버전의 GnuPG나 PGP를 가지고 있다면, 그 파일의 서명을 점검해야만 한다.(서명 부분을 보기 바란다).

3.2. 설정

여러분은 데비안 패키지나 RPM 패키지(레드햇 패키지 관리자) 혹은 소스 형태로 GnuPG를 구할 수 있다. GnuPG는 최신 레드햇 리눅스 배포판에 들어가 있다. 여러분의 시스템에 GnuPG가 설치되어 있는지를 확인하려면 다음 명령을 이용한다:

```
rpm -q gnupg
```

패키지는 리눅스 플랫폼에 필요한 도구와 함께 바이너리 파일 형태로 설치된다. 다른 플랫폼에서 사용할 GnuPG가 필요한 경우에는 여러분이 이것을 직접 컴파일하여야 한다. 여러분이 다른 플랫폼에서 사용할 수 있는 설치 방법을 일반 대중들이 알 수 있도록 한다면 고마운 일이다.

주요 부분에 대한 개발은 리눅스(x86)으로 수행하였기 때문에, 다른 시스템에 대한 변환작업을 문제로 보지 말아야 한다. GnuPG를 지원하는 것으로 알려진 운영 체제의 실제 목록은 GnuPG 홈페이지 (http://www.gnupg.org)에서 찾을 수 있다. 아래에 기술한 절차는 상당히 플랫폼 독립적이다. 이 절차를 이용하면 소스코드가 들어있는 타볼(tar-ball)로부터 GnuPG를 설치할 수 있다.

다음과 같은 명령을 이용하여 타볼을 푼다:

```
tar xvzf gnupg-?.?.?.tar.gz
```

압축을 푼후, 소스 코드가 들어가 있는 디렉토리로 들어간다. 그런 후 아래와 같이 입력한다.

```
./configure
```

위의 작업을 할때에 특별한 일(주로 에러)이 생기지 말아야 한다.

```
./configure --help
```

위의 명령을 이용하면 여러분은 컴파일 하는데 사용 가능한 설정 세팅을 볼 수 있다. 만약 국제화(GET text) 해야 하는데 있어 문제점이 생긴다면, **--with-included-gettext** 옵션을 이용하여 소스코드에 들어 있는 버전을 집어넣거나 **--disable-NLS** 옵션을 이용하여 전환할 수 있다.

3.3. 컴파일

위의 작업을 마친후 다음과 같은 명령어로 컴파일한다:

```
make
```

이것 역시 아무런 문제없이 작동해야 한다. 만약 문제가 발생하면 다음에 나와 있는 단계(여기서 기술하고 있는대로 같은 순서로)를 따르기 바란다: 우선 이 문제를 여러분 스스로 해결해 보려고 시도해라(물론 여러분이 구할 수 있는 문서를 이용하여). 그런 다음 여러분이 겪고 있는 문제가 알려진 버그가 아닌지 확인한다 (http://www.gnupg.org에 있는 BUGS 파일을 점검한다). 그런후에 여러분이 아는 누군가에게 질문한다. 만약 이런 방법으로 여러분의 문제를 해결하지 못한 경우 GnuPG-메일링 리스트에 질문을 올려본다 (정보의 출처 부분을 보도록 한다). 만약 문제가 경로와 관련된 것이라면, **make clean** 한 다음에, **configure**를 다시 실행하고 컴파일을 시도해야 한다. 이렇게 했는데도 안된다면 이제 겁먹을 시간이다.

3.4. 설치

자 이제 다음과 같이 입력하자:

```
make install
```

이 명령은 설치 디렉토리에 실제로 프로그램과 맨페이지를 복사한다. 만약 여러분이 ./configure를 실행할 때 설치 디렉토리를 변경하지 않았다면 **usr/local/share/gnupg/**가 설치 디렉토리가 된다. 이 설치 디렉토리는 라는 파일에서 찾을 수 있다. **options.skel** 파일 내에서 이 디렉토리를 찾을 수 있다. 여러분이 **options.skel** 파일을 변경하는 경우. 만약 여러분이 이것을 **~/.gnupg/options**에다 복사한다면 적절한 조절사항이 표준으로서 사용될 것이다. **~/.gnupg/**을

만들때 복사는 자동으로 되어야 한다. 모든 가능한 옵션은 문서로 잘 정리되어 있고 여기서 이를 설명하는 것은 유용할 것 같지 않다.

여러분은 GnuPG를 suid root로 실행할 수 있다. 그러면 프로그램은 수퍼유저가 가지고 있는 모든 권한으로 실행된다. 이렇게 함으로써 여러분은 프로그램의 특정 부분이 외부로 저장된 후 아무나 읽을 수 있도록 하는 가능성을 배제할 수 있다. 이러한 위험의 가치에 대해 필자더러 판단하라는 것은 현실적이지 못하다. 수퍼유저 권한으로 실행하는 트로이 목마는 시스템 전체를 손상시킬 수 있다. 만약 이러한 이유(혹은 다른 어떤 이유)로 여러분이 GnuPG를 루트 권한으로 실행하지 않도록 하겠다면 **~/.gnupg/options** 파일 내에 **no-secmem-warning**를 설정하여 경고 메시지를 끌 수 있다.

4. 키 이용하기

4.1. 키 만들기

다음 명령을 이용하여

```
gpg --gen-key
```

새로운 키쌍을 만든다. (여기서 키쌍이란 key pair로 비밀키와 공개키 두 가지를 쌍으로 묶어서 일컫는 것이다) 첫번째 질문은 어떤 알고리즘을 이용할 것인가다. 여러분은 PGP DH 대 RSA FAQ (http://www.hertreg.ac.uk/ss/pgpfaq.html) 혹은 응용된 암호화 방법에서 알고리즘에 대해 더 읽을 수 있다. 여러분은 손쉽게(아마 사용해야만 할 것이다 - 왜냐면 굉장히 널리 사용하고 있기 때문에) DSA/ElGamal을 이용할 것이다. 이것은 특허가 나 있지 않다.

다음 질문은 키의 길이다. 이것은 아주 사용자 의존적인 것이다. 여러분은 보안과 계산 시간 사이에서 선택을 할 필요가 있다. 키가 긴 경우에는 메시지가 샌 경우 크랙킹에 대한 위험은 줄어든다. 하지만 키가 크면 계산 시간 역시 증가한다. 계산 시간이 문제라면 여러분은 어떤 경우에 그 키를 이용하기를 원하는지 고려해야만 한다. 우리 모두는 연산 성능이 아주 빨리 올라가고 있다는 것을 안다. 새로운 프로세서는 갈수록 빨라지고 있다. 그러므로 이걸 새겨두자. GnuPG가 요구하는 최소한의 키 길이는 768비트이다. 하지만 어떤 사람들은 여러분더러 2048비트(현재로서 GnuPG에서 쓸 수 있는 최대이기도 하다) 길이의 키를 사용해야 한다고 말한다. DSA의 경우 1024가 표준 크기이다. 보안이 최우선이고 성능이 덜 중요하다면 여러분은 최대한 큰 키 크기를 선택하여야 할 것이다.

시스템은 이제 이름, 코멘트 그리고 이메일 어드레스를 넣어달라고 할 것이다. 여기 입력한 사항을 기반으로 하여 코드를 계산한다. 이 설정값은 나중에 변경할 수 있다. 키쌍 관리하기 부분을 보도록 하자.

마지막으로 여러분은 암호(사실 암호문장이 더 적절하다. 왜냐면 공백도 사용할 수 있기 때문이다)를 입력하여야 한다. 이 암호는 여러분의 비밀키에 속한 기능을 이용 가능하도록 하는데 사용한다. 좋은 암호문장은 다음과 같은 요소를 갖는다:

- 길고,
- 특수문자(알파벳과 숫자가 아닌)가 들어가 있으며,
- 특별한 것(이름이 아닌)이고,
- 추측하기 대단히 힘들어야 한다. (그러므로 이름, 생년월일, 전화번호, 크레디트 카드번호, 수표 계좌번호, 애들 숫자 등은 아니다)

어떤 경우에는 CaPItaLs와 같이 대문자를 섞어서 그리고 어떤 경우에는 SOMEtimes와 같이 하면 여러분은 보다 나은 보안환경을 만들 수 있다. 암호를 만들 때에는 여러분이 확실히 잊어버리지 않는 것으로 만들어야 한다. 만약 여러분이 암호를 잊어버리면 암호화된 메시지를 읽을 수 읽을 수 없고 여러분이 비밀키를 사용한 의미도 사라진다. 이러한 정보를 담은 인증서와 같은 것을 만들어 보는 것도 현명할 수 있다. (물론 어느 누구도 여러분의 암호문장을 입수하지 않도록 주의하여야 한다) 폐기 부분을 보기 바란다.

모든것을 입력한 후에 시스템은 키를 만들기 시작한다. 시간이 약간 걸린다. 이 시간동안 많은 무작위 데이터를 수집해야 한다. 다른 화면에서 작업함으로써 여러분은 시스템이 변하는 무작위 데이터를 수집하는 것을 도울 수 있다. 여러분이 지금까지 이해한 것처럼, 키는 항상 값이 다르다. 만약 여러분이 지금 키를 생성한 후 5분 뒤에 정확히 같은 데이터로 키를 성성해 보면 두 개의 키가 다르다는 것을 알 수 있다. 이제 왜 여러분이 여러분의 암호를 잊지 말아야 하는지를 이해하여야만 한다.

4.2. 키 내보내기

어떤 사용자에 대해 키를 내보내기 위한 명령은 다음과 같다:

```
gpg --export [UID]
```

UID를 지정하지 안은 경우에는 현존하는 모든 키를 내보낸다. 기본적으로 출력은 **stdout**(표준출력, 즉 화면)이다. **-o** 옵션을 이용하면 이를 파일로 내보낸다. 키를 바이너리 파일 대신에 7비트 아스키 파일로 쓰려면 **-a** 옵션을 이용하는 것이 좋다.

공개키를 내 보냄으로써 여러분은 여러분의 활동범위를 키울 수 있다. 다른 사람들이 안전하게 여러분과 접촉을 시작할 수 있다. 여러분이 여러분의 홈페이지, 핑거 서비스, http://www.pca.dfn.de/dfnpca/pgpkserv/와 같은 키 서버 혹은 여러분이 생각하는 어떠한 방법을 이용해서 공개키를 누구나 볼 수 있도록 공개하면 된다.

4.3. 키 가져오기

여러분이 어떤 사람의 공개키(혹은 여러개의 공개키)를 받은 경우 이를 사용할 수 있도록 여러분의 키 데이터베이스에 이들을 추가해야만 한다. 다른 사람의 공개키를 데이터베이스로 가져오는 명령어는 다음과 같다:

```
gpg --import [파일명]
```

만약 파일명을 생략하면 키는 **stdin**(표준 입력)을 통해 받게 된다.

4.4. 키 폐기하기

서너가지 이유 때문에 여러분은 현재 있는 키를 폐기하고자 하는 경우가 생길 수 있다. 예를 들어, 비밀키를 분실하거나 비밀키가 다른 사람 손에 들어간 경우, UID가 변경된 경우, 더 이상 그 키는 충분히 크다고 할 수 없는 경우 등이다. 이러한 모든 경우에 해당 키를 폐기하는 명령은 다음과 같다:

```
gpg --gen-revoke
```

이 명령은 폐기 인증서를 생성한다. 이를 할 수 있도록 하기 위해서는**,** 여러분은 비밀키가 필요하고**,** 다른 사람도 여러분의 인증서를 폐기할 수 있다. 이것은 하나의 단점을 가진다. 만약 자신이 암호문장을 모르면 키는 쓸모가 없게 된다. 하지만 자신 역시 키를 폐기할 수 없다! 이러한 문제를 극복하기 위해서 여러분이 키쌍을 생성할 때 폐기 라이선스를 만드는 것이 현명하다. 그리고 여러분이 그렇게 한 경우에 라이선스를 안전하게 보관해야 한다! 보관은 디스크, 종이 등과 같은 것에 할 수 있다. 이 인증서가 확실히 다른 사람 손에 들어가지 않도록 해야만 한다!!! 이를 허술하게 보관하는 경우 제 3자가 여러분의 키에 대한 폐기 인증서를 발행하여 키를 쓸모없게 만들 수 있다.

4.5. 키 관리

GnuPG 시스템에는 마치 데이터베이스 역할을 하는 파일이 들어있다. 이 파일에는 키와 관련한 모든 정보와 키에 대한 모든 데이터가 저장된다. (소유자 신용도 값 - Ownertrust 값까지 모든 것까지: 이와 관련한 더 자세한 정보는 키 서명하기 부분을 읽기 바란다) 다음 명령을 이용하면

```
gpg --list-keys
```

현존하는 모든 키를 볼 수 있다. 서명까지 보려면 다음과 같이 입력한다:

```
gpg --list-sigs
```

(더 자세한 내용은 키 서명하기 부분을 보기 바란다). 지문을 보고자 한다면 다음과 같이 입력한다:

```
gpg --fingerprint
```

여러분이 어떤 누군가가 정말로 그들이 주장하는(전화통화에서처럼) 그 사람인지를 확인하기 위해 "지문"을 보길 원한다. 이 명령어는 상대적으로 작은 숫자의 목록을 결과로 보여준다. 비밀키의 목록을 보고 싶다면 다음과 같이 입력한다:

```
gpg --list-secret-keys
```

비밀키에서 뽑아낸 지문과 서명을 목록으로 보는 것은 어떻게 하든간에 쓸데없는 행동이라는 것에 주의하기 바란다.

공개키를 지우기 위해서는 다음과 같이 입력한다:

```
gpg --delete-key UID
```

비밀키를 지우기 위해선 다음과 같이 입력한다:

```
gpg --delete-secret-key
```

키를 가지고 작업하는 데 관련한 중요한 명령어가 하나 더 있다.

```
gpg --edit-key UID
```

이 명령을 이용하여 여러분은 키의 유효 일자(다른 것들 가운데서), 지문 추가, 키 서명을 할 수 있다. 비록 너무 논리적이어서 언급하기에 뭐하지만. 이를 위해서는 여러분은 여러분의 암호문장이 필요하다. 이 명령을 입력하면 여러분은 명령행을 보게 될 것이다.

4.6. 키 서명하기

앞서 서론에서 언급했다시피 이 시스템에는 중요한 아킬레스건이 하나 있다. 바로 공개키의 신빙성 문제이다. 만약 여러분이 잘못된 공개키를 갖는 경우 여러분은 여러분의 암호화 값에 빠빠이를 할 수 있다. 그러한 위험을 극복하기 위해 서명하는 키에 대한 여지가 있다. 그러한 경우 여러분은 그 키에 대해 서명을 넣을 수 있고, 이렇게 하면 이 키는 확실히 맞는 것이구나라고 확신할 수 있다. 이러한 방법은 서명을 통해 키에서 언급한 사용자 신분이 실제로 그 키의 소유자라는 것을 알려주는 상황으로 이끌어간다. 그렇게 재확인함으로써 여러분은 암호화를 시작할 수 있다.

서명이 필요한 해당 키에 대해 **gpg --edit-key UID** 명령을 이용하므로써 여러분은 **sign** 명령으로 서명할 수 있다.

여러분은 여러분이 그 키가 정말로 믿을 수 있는 것이라고 절대적으로 확실한 경우에 확실한 키라고 서명하여야 한다**!!!** 그러므로 여러분이 확신한 경우 여러분은 키를 스스로 갖거나(마치 키 서명 단체에서 처럼) 다른 수단을 통해 키를 입수한 경우에는 지문-메카니즘을 이용하여 해당 키를 확인하기(예를 들어 전화를 이용하여) 바란다. 어떤한 경우라도 절대 추측해서 키에 서명해서는 안된다.

현재 가용한 서명과 "소유자 신용도"를 기초로 하여 GnuPG는 키의 유효성을 결정한다. 소유자 신용도란 하나의 값으로 키의 소유자가 어떤 특정 키에 대한 신용도 수준을 결정하는 데 이용하는 것이다. 그 값은 다음과 같다.

- 1 = 알지 못한다
- 2 = 신용하지 않는다
- 3 = 간신히 신용한다
- 4 = 완전히 신용한다

만약 사용자가 서명을 신용하지 않는 경우 그 서명은 무시한다고 할 수 있다. 신용도 정보는 키와 동일한 파일에 저장하지 않고, 따로 다른 파일에 저장한다.

5. 암호와 복호

모든 것을 설치하고 원하는 대로 설정을 한 후, 이제 암호화와 복호화를 시작할 수 있다.

암호화나 복호화를 할때는 1개 이상의 비밀키를 사용할 가능성이 있다. 이런 상황이 벌어지는 경우 여러분은 활성키를 선택할 필요가 있다. 이것은 **-u UID** 옵션이나 **--local-user UID** 옵션을 이용하여 할 수 있다. 이렇게 하면 기본키를 원하는 키로 대체하게 된다.

받는 사람을 수정하고픈 경우에는 **-r** 옵션이나 **--recipient** 옵션을 이용해서 할 수 있다.

5.1. 암호

암호화 하는 명령은 다음과 같다:

```
gpg -e 받는사람 [데이터]
```

혹은

```
gpg --encrypt 받는사람 [데이터]
```

제 3자가 당신의 신분을 위장하는 위험을 피하기 위해서, 여러분이 암호화하는 모든 것에 서명하는 것이 아주 유용하다. 서명 부분을 보기 바란다.

5.2. 복호

복호화 하는 명령은 다음과 같다:

```
gpg [-d] [데이터]
```

혹은

```
gpg [--decrypt] [데이터]
```

또한 여기서 **stdout**이 기본값이지만, **-o** 옵션을 이용하여 출력을 파일로 할 수도 있다.

6. 서명하고 서명 확인하기

여러분 자신의 키로 데이터를 서명하기 위해선, 다음과 같은 명령을 이용한다:

```
gpg -s (or --sign) [데이터]
```

이렇게 하면 동시에 압축도 행한다. 이는 결과물을 판독하기 쉽지 않다는 것이다. 여러분이 판독하기 쉬운 결과물을 원한다면 다음과 같은 명령을 쓸 수 있다:

```
gpg --clearsign [데이터]
```

이렇게 하면 확실히 결과물을 읽기가 수월하다. 더 나아가 (서명한 데이터)에 대해서도 똑같은 형태의 결과물을 얻을 수 있다.

다음 명령을 이용하여

```
gpg -b (or --detach-sign) [데이터]
```

여러분은 서명을 따로 다른 파일에 쓸 수 있다. 바이너리 파일(예를 들어 압축파일과 같은)을 서명할 때 이 옵션을 이용하도록 강력히 추천한다. 또한 **--armor** 옵션은 이와 같은 상황에서 엄청나게 유용하다.

여러분은 상당히 자주 데이터가 서명 뿐만 아니라 암호화 되어 있는 경우를 볼 수 있다. 명령어를 전부 다 적으면 다음과 같이 보인다:

```
gpg [-u 보내는사람] [-r 받는사람] [--armor] --sign --encrypt [데이터]
```

옵션 **-u** (**--local-user**)와 **-r** (**--recipient**)의 기능은 앞서 기술한 바와 같다.

암호화 된 데이터가 서명까지 되어 있는 경우, 서명은 데이터를 복호화 할때 확인된다. 여러분은 다음의 명령을 이용하여 서명한 데이터의 서명을 확인할 수 있다:

```
gpg [--verify] [데이터]
```

이것은 (물론) 여러분이 보낸 사람의 공개키를 가지고 있는 때에만 작동할 것이다.

7. 프런트 엔드

GnuPG를 아주 손쉽게 이용할 수 있도록, 여러분은 GnuPG 암호화를 이용하거나 지원하는 프로그램을 폭넓게 선택할 수 있다. 여러분의 키를 마우스 버튼 클릭으로 관리하는 그래피컬 프런트 엔드가 있고 많은 MUA(Mail User Agents)를 이용하면 여러분은 이메일 메시지를 매끈하게 암호화하고 서명할 수 있다. 프런트 엔드에 대한 거의 대부분의 목록은 GnuPG 프런트엔드 (http://www.gnupg.org/frontends.html) 페이지에서 찾을 수 있다. 우리는 이 부분에서 몇 가지를 강조하고자 한다.

7.1. 그래피컬 인터페이스

7.1.1. GPA

GPA (http://www.gnupg.org/gpa.html)라는 것은**GNU Privacy Assistant(**번역하면 **GNU** 사생활 도우미)라는 것으로, GnuPG용으로 사용할 수 있는 그래피컬 유저 인터페이스(GUI)이다. 이것은 표준 그래피컬 프런트 엔드로서, GnuPG 프로젝트에서 제공한다. GPA를 이용하면 여러분은 여러분의 키링을 볼 수 있고, 키를 내보내고 가져오고, 키를 생성하고, 키의 정보를 편집하고 또한 문서를 암호화, 서명, 복호화 할 수 있다. GPA를 설치하는 것은 쉽다. 타볼을 다운로드 하여 압축을 푼 다음에 늘 하는 식으로 아래와 같이 하면 된다.

```
./configure; make; make install.
```

다음과 입력하면 프로그램을 시작할 수 있다.

```
gpa
```

7.1.2. Seahorse

Seahorse (http://seahorse.sourceforge.net/)는 GnuPG와 함께 쓸 수 있는 GNOME 프런트엔드다. 이것을 이용하면 텍스와 파일을 서명, 암호화, 확인 그리고 복호화할 수 있다. 텍스트는 클립보드에서 가져오거나, Seahorse에 들어 있는 작은 편집기에 직접 써 넣을 수 있다. Seahorse는 또한 키 관리자이기도 하다. 즉 여러분의 키링에 저장된 거의 대부분의 키 정보를 수정하는데 사용할 수 있다. Seahorse는 데비안 패키지를 가지고 설치할 수도 있다. (현재까지 RPM은 없다) 아니면 소스 타볼을 이용하여 설치할 수도 있다. 소스를 이용하여 설치하는 것은 다른 패키지에서 처럼 비슷하다. 다운로드해서, 타볼을 푼 다음, configure하고 make install 하면 된다. 기본적으로 설치하면 seahorse는 **/usr/local** 디렉토리에 들어가고 Gnome 'Application' 메뉴에 메뉴 항목으로 나온다.

7.1.3. Geheimnis

Geheimnis (http://geheimnis.sourceforge.net/)는 KDE를 기반으로 하는 GnuPG용 프런트 엔드다. GPA 혹은 Seahorse에 있는 모든 기능을 가지고 있다는 것을 전제로 한다. 하지만, 필자가 프로그램을 컴파일 하려고 했을때 라이브러리(/usr/lib/libfam.la)가 없다는 에러가 났다.

7.2. 이메일 프로그램

대부분의 인기있는 이메일 프로그램(혹은 MUAs)는 GnuPG를 지원한다. 이들 가운데 최소한의 것을 골라본다면 다음과 같다:

- Mozilla
- Pine
- Kmail
- Eudora
- Mutt
- exmh

아마 더 있을 것이다; 전부 다 되는지 안되는지 일일이 해 본다는 것은 불가능한 일이다.

여러분의 메일 프로그램 내에서 GnuPG 지원을 이용하면 여러분은 여러분의 공개키로 암호화 되어 여러분에게 온 이메일 메시지를 복호화 하고 여러분의 메시지를 서명할 수 있다. 그래서 받는 측에서는 여러분이 메시지의 저작자인 것과 여러분의 메시지를 받는 사람의 공개키로 여러분의 이메일을 암호화 한다는 것을 확신할 수 있는 것이다.

7.2.1. Mozilla와 Enigmail

모질라는 자체에서 GnuPG를 지원하지 않는다. GnuPG 암호화를 모질라와 함께 사용하려면 여러분은 EnigMail (http://enigmail.mozdev.org/)과 같은 플러그-인을 설치하여야 한다. Enigmail은 모질라/네스케이프 메일용 "플러그-인"으로서 사용자가 유명한 GPG와 PGP 소프트웨어가 제공하는 승인과 암호화 기능을 사용할 수 있도록 해주는 역할을 한다. Enigmail은 메일을 보낼때 암호화/서명을 할 수 있고, 또한 메일을 받을 때 복호화/승인을 할 수 있다. 또한 공개키를 가져오기/내보내기 할 수도 있다.

RedHat 리눅스 시스템에 EnigMail을 설치하는 것은 특별히 쉽다. 왜냐면 최신 버전의 모질라용 RPM이 있기 때문이다. 설치할 패키지는 다음과 같이 두 가지이다:
mozilla-enigmail-0.39-3.i386.rpm와 **mozilla-ipc-0.99-0_rh7x.rpm**이 그것이다. 이들 RPM을 설치한 후, 모질라(혹은 네스케이프 6.x)를 다시 시작해야 여러분의 이메일에서 GnuPG를 사용할 수 있다. 하지만, EnigMail은 단지 여러분의 메시지에 있는 텍스트만을 암호화 한다는 사실에 주의하기 바란다. 첨부한 내용은 EnigMail이 암호화 하지 않는다. 즉 여러분이 따로 첨부하고자 하는 파일은 미리 암호화 되어 있어야 한다. 다른 시스템에서는 소스코드를 이용해서 EnigMail을 설치하여야 할 것이다.

7.2.2. Kmail

Kmail은 KDE용 표준 이메일 프로그램으로, GnuPG와 PGP 암호화 지원 기능이 통합되어 있다. 여러분이 서명하고 복호화 할 수 있도록 설정하기 위해서는, 여러분은 Kmail 설정 내에 'Identity' 섹션에서 여러분의 GnuPG 사용자 신분(user ID)를 입력하여야만 한다. 여러분이 새로운 메시지를 보내는 경우, 기본적으로 메시지는 서명되거나 암호화 되지 않는다. 여러분은 툴바에 있는 'Sign message'와 'Encrypt message' 버튼을 체크 해 주어야만 한다.

8. 정보의 출처

8.1. GnuPG

- GnuPG 홈페이지 (http://www.gnupg.org)
- GnuPG 홈페이지 (http://www.gnupg.org/docs.html)에 대한 GnuPG 메일링 리스트와 문서, 설명 포함.
- GnuPG 프로젝트에 함께 들어가 있는 정보(버전 0.9.2까지 수정됨)는 아직까지는 상세하게 작업이 되어 있지 않다. 그리고 잊지 말 것은 이것이다:

  ```
  gpg --help
  ```

이것은 대단히 중요한 정보다.

8.2. PGP

PGP는 더 오래되고 (여전히) 널리 퍼져 사용되고 있는 암호화 프로그램이다. 수년간에 걸쳐 많은 문서가 만들어졌다. 이것은 대한히 유용한 정보로 간주할 수 있을 것이다. 그러한 정보 중 많은 것은 상당히 일반적인 것으로 GnuPG에도 적용할 수 있다. 이들 문서를 보고 싶다면 다음에 나와 있는 URL을 참조하기 바란다:

- 국제 PGP 홈페이지 (http://www.pgpi.com)
- PGP DH 대 RSA FAQ (http://www.hertreg.ac.uk/ss/pgpfaq.html) 문서는 두가지 알고리즘의 차이점에 대한 정보를 담고 있다. 이 두가지는 GnuPG에서 모두 사용하는 것이다.

8.3. 키 서버

- Keyserver.net (http://www.keyserver.net)
- http://wwwkeys.eu.pgp.net

8.4. 서적

- B. Schneier, "Applied Cryptography, Second Edition", Wiley, 1996 Deutsche Ausgabe unter dem Titel "Angewandte Kryptographie", Addison-Wesley, 1996